(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 100 018 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
16.05.2001 Bulletin 2001/20

(51) Int Cl.7: **G06F 12/06**

(21) Application number: **00310046.8**

(22) Date of filing: **10.11.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **11.11.1999 JP 32042699**
12.11.1999 JP 32295599
12.11.1999 JP 32226799
24.11.1999 JP 33347699

(71) Applicant: **SANKYO SEIKI MFG. CO. LTD.**
Suwa-gun, Nagano-ken 393-8511 (JP)

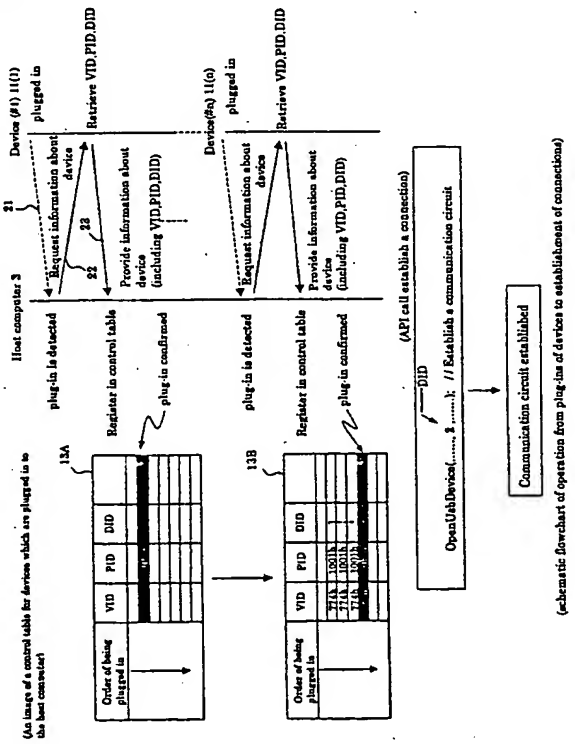
(72) Inventors:
- **Komatsu, Tetsuro, Sankyo Seiki Mfg. Co., Ltd.**
Suwa-gun, Nagano, 393-8511 (JP)
- **Obinata, Eiichi, Sankyo Seiki Mfg. Co., Ltd.**
Suwa-gun, Nagano, 393-8511 (JP)

(74) Representative: **Kinsler, Maureen Catherine et al**
Kilburn & Strode,
20 Red Lion Street
London WC1R 4PJ (GB)

(54) USB-Interface Equipped Device

(57) A method for communication control of a USB-equipped device that has a vendor identification (VID), a product identification (PID) and a device identification number (DID) that is different from the VID and PID. The method comprises transmitting the DID, together with the VID and the PID to a host computer when the USB-equipped device is plugged in; identifying in the host computer the model of the USB-equipped device based on the transmitted VID and PID and identifying a USB-equipped device as a target of communication among a plurality of USB-equipped devices using the DID transmitted, thereby to established communication with said USB-equipped device.

Fig.4



EP 1 100 018 A2

Description

[0001] A Universal Serial Bus (USB) interface is known as one of the interfaces to connect a computer and peripherals. Fig. 7 shows a schematic configuration of a system connected via the USB. Herein, USB-equipped device 1 to be controlled is connected to host computer 3 via USB interface 2. USB-equipped device 1 to be controlled is comprised of CPU 4, memory 5, and USB controller 6. A communication circuit can be created between USB-equipped device 1 and host computer 3 via USB interface 2 and USB controller 6. Based on commands from host computer 3, USB-equipped device 1 performs a given control operation. Memory 5 of USB-equipped device 1 contains a VID and PID which will be needed for host computer 3 to specify the device.

[0002] USB-equipped device 1, which carries out communication with host computer 3, transmits the VID and PID to host computer 3 during the plug-in process (establishing a physical connection). With host computer 3, a hardware device driver specific to the plugged-in device, which is specified by the VID and PID, is loaded and built into the operating system. As a result, the operating system is prepared to establish a connection with the device by executing the application program. When the application program is executed, the loaded hardware device driver is called such that a communication circuit is created with the connected device.

[0003] Fig. 8 shows the processing operation when a device is connected. As shown in the figure, when multiple devices are plugged in, the VID and PID of each device are stored in order of being plugged in within control table 7 in memory 5 of host computer 3. Therefore, host computer 3 can create a communication circuit by specifying a device as a target of communication based on the stored VIDs and PIDs.

[0004] The configuration of the USB does not limit the plug-in process of multiple devices of the same product (or model), in other words, devices having the same VID and PID. However, a user's application program may not have a means effective in identifying multiple devices of the same product (or model) which are plugged in to the host computer. As a result, the host computer cannot distinguish one device from another, causing the possibility that a connection may not be established with a device as a target of communication.

[0005] Naturally, those devices may be controlled by means of a control table in which the devices are aligned in order of being plugged in, as shown in Fig. 8. Nonetheless, the order of the devices of being plugged in may be changed afterward (in other words, there may be a change in the position of the device of being connected to the host computer via a USB hub). In this case, the host computer cannot identify the relative positions of the devices being connected. Therefore, it is not realistic to identify multiple devices of the same product (or model) according to the order of being plugged in.

[0006] In order to solve the above issue, it is suggested to assign a different PID to each device even though devices are of the same product (or model), for example. According to this method, the devices are recognized as different devices although for users they are of the same product (or model) in terms of the software. However, this is not desirable as maintenance of devices become extremely complex for both vendors and users.

[0007] The purpose of the present invention is, considering the above issue, to provide a method for communication control of devices, which are connected to a host computer via a USB interface, wherein it distinguishes multiple devices of the same product (or model).

[0008] The purpose of the present invention is also to provide a USB-equipped device which is suitable for the use of such a method.

[0009] Also, in general, the above mentioned USB-equipped devices have Plug and Play (PnP) enabling an automatic system configuration. Therefore, a user neither needs to change settings of the host computer to correspond to peripherals nor needs to reboot the host computer. When the USB-equipped device is connected to the host computer, as shown in Fig. 15, the host computer detects the USB-equipped device being connected when an electric change is detected (Step 11), followed by electric communication (hereafter, "USB communication") between USB terminals (Step 12). After confirming the type of the USB-equipped device, the host computer calls a corresponding driver to establish a system for USB communication. The communication with the USB-equipped device is repeated as necessary as long as the communication is enabled (Step 13).

[0010] On the other hand, when the host computer detects an error in the USB communication, the host computer can try to restore the line to the USB-equipped device with a communication error, only if the host computer has a USB control program capable of doing so. For example, when an error occurs in the USB communication, the USB control program of the host computer retries the PnP after a bus reset. If the communication is restored by the trial, the system may be reestablished.

[0011] However, the host computer is configured such that the USB communication is established from the host computer side, but not from the USB-equipped device side. When the host computer determines that there is a communication error and that the USB-equipped device is the source of the error, the host computer may try to restore the communication with the USB-equipped device. In contrast, when the USB-equipped device determines that the communication error is caused by the host computer, the USB-equipped device cannot start the communication until the host computer restores the communication by some means (e.g. someone reboots the host computer).

[0012] When the host computer cannot detect the communication error caused by the USB-equipped device, the USB-equipped device detects its own error by

a watchdog timer such that the USB-equipped device may try to restore the communication by reinitializing itself including the USB communication. However, the USB communication is a one-way process from the host computer to the USB-equipped device. Therefore, the USB-equipped device cannot start the communication with the host computer or initialize the USB communication. In other words, in spite of the possibility of the USB-equipped device to restore the USB communication, the USB-equipped device cannot actively initiate any means to restore the USB communication.

[0013] In this case, it is known that the USB communication can be artificially restored, although it is not a complete restoration, by removing and reconnection of the USB cable terminal from / to the USB connector of the USB-equipped device. However, it is very inefficient to remove and reconnect the cable terminal every time a communication error occurs. Additionally, it is impossible to perform the removing and reconnection of the terminal in unattended operation.

[0014] Therefore, the present invention intends to provide a method for restoring communication with a USB-equipped device and an apparatus to restore communication with the same in which communication errors between a host computer and a USB-equipped device can be detected wherein the USB-equipped device can perform a measure to restore the communication in relation to the host computer.

[0015] A Universal Serial Bus (USB) interface is known as one of the interfaces to connect a computer and peripherals. When a USB-equipped device, such as a mouse or a key board, is connected to a USB terminal of a host computer via a USB cable, the host computer detects an electric change in the USB interface to confirm that the USB-equipped device is connected. The host computer electrically communicates with the connected USB-equipped device (hereafter "USB communication") and confirms the kind of the USB-equipped device based on the response. Thereafter, a USB-equipped device driver corresponding to the kind of the USB-equipped device is loaded such that the USB-equipped device can be controlled, resulting in a plug and play (hereafter "PnP") method in which a USB-equipped device becomes useable by connecting the USB-equipped device to a host computer by a USB cable.

[0016] The USB communication can be performed by using two lines, D+ signal line and D- signal line. A USB cable also has a power supply line, which enables power supply from the host computer, and a ground line. In other words, the USB cable and the USB terminal have the above four lines.

[0017] There are two methods for power supply with USB-equipped devices. First, power can be supplied from the USB terminal of the host computer via the power supply line of the USB cable (hereafter "bus-powered"). Another method is that the USB-equipped device has its own power supply (hereafter "self-powered"). This self-powered USB-equipped device, having its own power supply, utilizes a commercial power supply, for example, such that a power supply from the host computer is not necessary.

[0018] However, the USB-equipped device cannot detect the connection of the USB cable with either methods of power supply. Nonetheless, the bus-powered USB-equipped device loses the power supply once the connection of the USB cable is discontinued (e.g. the cable is physically pulled out). As soon as the connection of the cable is discontinued, all the operation is suspended. In other words, even though the connection of the USB cable cannot be detected, the operation is suspended as soon as the connection is discontinued. Therefore, detection of the connection is not necessary.

[0019] However, with the above USB-equipped device having its own power supply, the power supply will be continued after the connection of the USB cable is discontinued such that the operation proceeds. If the connection of the USB cable is detected, a measure to suspend the operation may be performed. However, the USB-equipped device cannot detect the connection of the USB cable such that no measure can be performed to suspend the operation.

[0020] In the case of the self-powered USB-equipped device, the operation continues even after the connection with the host computer is ended. In other words, the operation continues even when the USB-equipped device is no longer under the control of the host computer.

[0021] Other than the time when the USB cable is pulled out, the undesired continuity of the operation as described above may occur when the connection between the host computer and the USB-equipped device is physically discontinued, e.g. the contact failure of the USB connector and the broken USB cable.

[0022] The purpose of the present invention is to provide a USB-equipped device which can detect physical discontinuity of the connection between a host computer and the self-powered USB-equipped device.

[0023] Some types of USB cables include USB cable 401 having USB plugs on both ends as shown in Figure 23(A) and USB cable 402 which is directly derived from USB-equipped device 403 and has a USB plug at one end as shown in Fig. 23(B).

[0024] It is known that if the host is correctly operating it can reconfigure a USB-equipped device by removing and reconnecting a USB cable.

[0025] However, by repeatedly removing and reconnecting the USB cable, the contact of the connector is worn by friction, and further, connection failure of the connector may result. In addition, the connector may be damaged when someone removes and reconnects the USB cable. The connection failure and damage of USB cable 401 having USB plugs on both ends are not largely problematic as USB cable 401 is not very expensive such that it can be easily replaced. However, in the case of USB cable 402 which is derived from USB-equipped

device 403, USB cable 402 as a part of USB-equipped device 403 needs to be replaced or repaired, resulting in drawbacks such as a high repair cost and inconvenience as the device cannot be utilized during the repair. Further, it is believed that the frequency of removing and reconnection of the USB cable is low in general use. However, during development of a USB-equipped device, removing and reconnection of the USB cable may be frequently performed for reasons such as the USB-equipped device under development does not operate, resulting in the above connection failure and damage.

[0026] Therefore, the present invention intends to provide a USB cable which can simulate a condition as if a USB cable is removed and reconnected without an actual physical action such that wearing and damage of a USB connector portion can be avoided.

[0027] Various aspects of the present invention are specified in the independent claims. Some preferred features are set out in the dependent claims.

[0028] More specifically, a first aspect of the invention provides a USB-equipped device which is given vendor identification and product identification to communicate with a host computer via a USB interface wherein a device identification different from said vendor identification and said product identification is given. Further provided is a method for communication control of said USB-equipped device in which said USB-equipped device is given said device identification by a device such as rotary switch, dip switch, non-volatile memory and the like.

[0029] Also provided a method for communication control of said USB-equipped device which establishes a connection from said host computer to said USB-equipped device via said USB interface wherein when said USB-equipped device is plugged in, it transmits said device identification, together with said vendor identification and said product identification, to said host computer; said host computer identifies the model of said USB-equipped device based on said transmitted vendor identification and product identification; and if a plurality of said identified USB-equipped devices are plugged in, a USB-equipped device as a target of communication is identified among said plurality of USB-equipped devices based on said sent device identification such that a connection is established with said USB-equipped device.

[0030] Preferably, the host computer assigns said device identification given to said USB-equipped device as a target of communication by giving said assignment to said USB-equipped device as one of the arguments which are called by an application program interface when an application program is executed.

[0031] Preferably, said host computer stores said vendor identification, said product identification and said device identification of said plugged in USB-equipped devices in order of being plugged in a form of a control table; a connection is established to each device sequentially as recorded in said control table; and when said device identification sent from said connected USB-equipped device matches a device ID given by an argument of said application programming interface, it is determined that connections to said USB-equipped devices as targets of communication are established.

[0032] According to yet a further aspect of the invention, there is provided a method for restoring communication with a USB-equipped device which is connected to a host computer via a USB connector wherein said USB-equipped device has an error detecting means, which detects an error in communication with said host computer, and a voltage changing means which changes the voltage of a signal line connected to said USB connector of said USB-equipped device; and when said error detecting means detects an error in the communication with said host computer, said voltage changing means changes the voltage of said signal line such that a condition, where said USB connector is reconnected to said host computer, is simulated.

[0033] Preferably, the voltage changing means has one of the following: a switch between a power supply and said signal line; a switch between the ground and said signal line; or a switch to open and close said signal line.

[0034] According to yet another aspect of the invention, there is provided an apparatus to restore communication with a USB-equipped device which is connected to a host computer via a USB connector wherein said USB-equipped device has an error detecting means, which detects errors in communications with said host computer and a voltage changing means which changes the voltage of a signal line connected to said USB connector of said USB-equipped device; and when said error detecting means detects an error in the connection with said host computer, said voltage changing means changes the voltage of said signal line such that a condition, where said USB connector is reconnected to said host computer, is simulated.

[0035] According to still another aspect of the invention, there is provided a USB-equipped device which is configured such that it is connected to a host computer via a USB cable, having a signal line and power supply line to communicate with said host computer and that it has its own power supply wherein said USB cable is connected to a connector portion of said USB-equipped device; a voltage detecting portion is connected to a terminal for said power supply line at said connector portion to detect the power supply voltage of said host computer; and a connection of said USB cable can be detected based on the output from said voltage detecting portion.

[0036] Preferably, the USB cable has said signal line comprised of a D+ signal line and a D- signal line, said power supply line and a ground line wherein said connector portion of said USB-equipped device has connection terminals corresponding to each line.

[0037] The USB-equipped device may operate using said individual power supply wherein said voltage de-

tecting portion in said USB-equipped device provides the output indicating whether a given voltage exists to a control portion of said USB-equipped device such that when disconnection of said USB cable is detected, the operation of said USB-equipped device is suspended.

[0038] According to a still further aspect of the invention, there is provided a USB data cable having a USB plug which connects a host computer with a USB-equipped device wherein a signal line in said USB data cable having said USB plug has a switch to generate an electric change in said signal line; and said switch can be controlled externally.

[0039] According to a yet still further aspect of the invention, there is provided a USB data cable having a USB plug which connects a host computer with a USB-equipped device, receiving power supply from said host computer, wherein a signal line in said USB data cable having said USB plug has a switch to open and close a power supply line; and said switch can be controlled externally. The switch may turn on and off said signal line.

[0040] The USB data cable may have a D+ signal line and a D- signal line wherein said switch generates an electric change in said D+ signal line and said D- signal line.

[0041] Preferably, the switch is formed in a housing of said USB plug and has an operating portion which to be pressed.

[0042] Various aspects in which the invention is embodied will now be described, by way of example only, and with reference to the accompanying drawings of which:

Fig. 1 is a schematic configuration of a system in which the present invention is embodied.

Fig. 2 is a schematic view of an API to which an argument is given to assign a USB-equipped device identification of a USB-equipped device in the system of Fig. 1.

Fig. 3 is a system in which multiple USB-equipped devices of the same product (or model) are connected to a host computer via a USB interface.

Fig. 4 is a schematic flowchart of operation from plug-ins of USB-equipped devices to establishment of connections in the system of Fig. 3.

Fig. 5 is a system in which an identical USB-equipped device identification is mistakenly assigned to connected devices.

Fig. 6 is a configuration of a control table which is stored in the host computer in the system of Fig. 5.

Fig. 7 is a schematic configuration of a conventional system in which a host computer and USB-equipped devices are connected via a USB interface.

Fig. 8 is a flowchart showing operation to plug in USB-equipped devices in the system of Fig. 7.

Fig: 9 is a schematic circuit configuration of an apparatus to restore communication with a USB-equipped device.

Fig. 10 shows circuit connections in the vicinity of a USB connector.

Fig. 11 is a flowchart showing USB communication and the restoration process thereof.

Fig. 12 is a schematic circuit configuration of another embodiment of an apparatus to restore communication with a USB-equipped device.

Fig. 13 is a schematic circuit configuration of yet another apparatus to restore communication with a USB-equipped device.

Fig. 14 is a schematic circuit configuration of yet another apparatus to restore communication with a USB-equipped device of the present invention.

Fig. 15 is a flowchart showing a conventional restoration process for errors in the USB communication by the host computer.

Fig. 16 is a schematic configuration of an embodiment of a USB-equipped device according to the present invention.

Fig. 17 is a flowchart showing the operation of a CPU with the USB-equipped device.

Fig. 18 is a schematic configuration in which a low speed USB-equipped device is connected to a host computer via a USB data cable having a USB plug.

Fig. 19 is a schematic configuration of an example of the USB data cable having a USB plug.

Fig. 20 is a side view indicating when a switch is not pressed.

Fig. 21 is a side view indicating when the switch is pressed.

Fig. 22 is a schematic configuration in which a high speed USB-equipped device is connected to a host computer via a USB data cable having a USB plug.

Fig. 23 is a schematic configuration of a conventional USB data cable having a USB plug. Fig. 23(A) is a type having USB plugs on both ends. Fig. 23(B) is a type which is directly derived from the USB-equipped device and has a USB plug on one end.

[0043] The first embodiment relates to a USB-equipped device and a method for communication control thereof in which even when some of multiple USB-equipped devices connected to a host computer via a USB interface carry identical vendor identifications (VIDs) and product identifications (PIDs), a connection can be established by identifying those IDs.

[0044] Fig. 1 shows a schematic configuration of a USB-equipped device 11 that has CPU 4, memory 5, and USB controller 6, similar to the USB-equipped device in Fig. 7. Memory 5 stores a VID and PID of the device. In addition, device 11 of this embodiment has device identification (hereafter, "DID") setting apparatus 12: DID setting apparatus 12 is comprised of a rotary switch, dip switch, a non-volatile ROM and the like and gives a given DID to USB-equipped device 11.

[0045] When an application program for each USB-equipped device, stored in the memory (not shown in the figure) of host computer 3, is executed, an applica-

tion program interface (hereafter "API") is called. The API assigns a DID to a USB-equipped device as a target of communication. In other words, assignment of a DID is performed by giving the assignment to the USB-equipped device as one of arguments called by the API. A schematic configuration of such an API is shown in Fig. 2.

[0046] Suppose multiple USB-equipped devices of the above configuration are plugged in, as shown in Fig. 3. If the USB-equipped devices are the same product (or model) manufactured by the same vendor, VIDs and PIDs of device 11(1), device 11(2), and device 11 (n) (wherein the "n" indicates a positive integer) are identical. In this example, device 11(1) receives a DID "1", device 11(2) receives a DID "2", and device 11 (n) receives a DID "n", accordingly.

[0047] Fig. 4 shows the process of establishing a connection between host computer 3 and device 11(1) through device 11(n). For example, when device 11(1) is plugged in (arrow 21), host computer 3 transmits the USB-equipped device a request for transmitting IDs (arrow 22). Device 11(1) retrieves its VID and PID from memory 5 under the control of CPU 4. The USB-equipped device also retrieves its DID, which is "1" in this case, from the DID setting apparatus and transmits the IDs to host computer 3.

[0048] When host computer 3 receives the above information, its operating system loads a hardware device driver specified by the VID and PID as a part of its functionality. Control table 13, in which VIDs, PIDs and DIDs are stored in order of devices being plugged in, is developed in the memory. Accordingly, the plug-in process of device 11(1) is completed. Similarly, the plug-in process of device 11 (n) is completed, such that control table 13B is developed in the memory of host computer 3.

[0049] Host computer 3 then executes the application program during establishment of a connection with each device. By calling the hardware device driver, the product (or model) of the device as a target of communication (VID and PID) is specified. In this example, as shown in Fig. 3, devices of "n" number of the same model, devices 11(1) through 11(n) are plugged in wherein their hardware device drivers are the same. Therefore, the device as a target of communication cannot be specified.

[0050] According to a DID given by an argument of the API, device as a target of communication is specified from devices of "n" number to establish a connection. In Fig. 4, a configuration of API 14, which establishes a connection by specifying device 11(2) having a DID of "2", is indicated in a box.

[0051] In general, the plug-in process of multiple USB-equipped devices having identical VID and PID is rare. Therefore, a connection can be established with USB-equipped devices according to the order of control table 13B, for example, and confirm whether the DID of the USB-equipped device, with which a connection is established, matches the DID given by an argument of the API. If they match, it is confirmed that a connection with a USB-equipped device as a target of communication is established such that the control shifts to the next process. If they do not match, the same process is repeated until a connection is established with a USB-equipped device carrying a matching DID.

[0052] As shown in Fig. 5, an error in setting the DID for devices 11(1) through 11 (n) of the same model may cause assignment of a DID "2" to two devices 11(2) and 11(n), for example. In this case, the control table may look like the one in Fig: 6. There are two major methods to solve the above situation:

(1) to establish a connection by considering a USB-equipped device, which meets the condition first during the process of specifying the USB-equipped device as a target of communication, that is a USB-equipped device having matching VID and PID, as a USB-equipped device as a target of communication; and

(2) to give up establishment of a connection by notifying an operator that there is an error in the setting of the DID by some means, such as showing an error message.

[0053] When only one USB-equipped device of the product (or model) is plugged in, the process of checking the DID can be omitted from the operation to specify a USB-equipped device as a target of communication.

[0054] Such a control form can be established by modifying the specifications of the application software of the host computer which controls the USB-equipped devices.

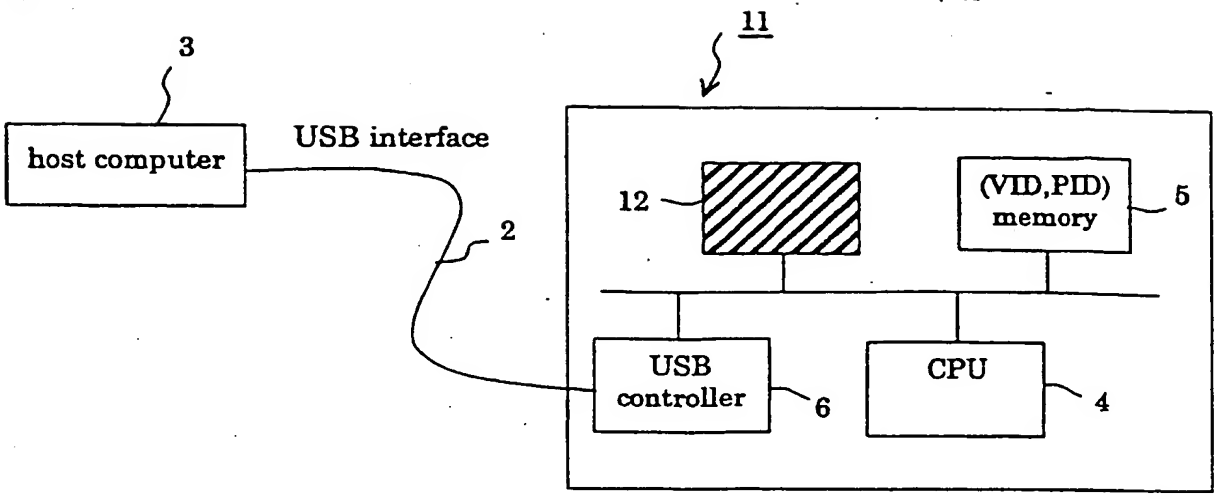
[0055] By giving a DID to each USB-equipped device, USB-equipped devices having the same VID and PID can be distinguished based on the DID. Therefore, the host computer always specifies a USB-equipped device to be controlled via the USB interface such that a desired communication control can be performed.

[0056] Further, a DID is given to each USB-equipped device, therefore, even when USB-equipped devices are of different models, the same hardware device driver can be assigned for USB-equipped devices which share a basis of a 'communication protocol. In other words, USB-equipped devices of different products (or models) can be handled as USB-equipped devices of the same product (or model).

[0057] The second embodiments relate to a method for restoring communication with a USB-equipped device and an apparatus to restore communication therewith. More specifically, the present invention relates to improvement in operation of communication failure between a host computer and a USB-equipped device.

[0058] Figs. 9 through 11 show that USB connectors 103, 103 of host computer 101 and USB-equipped device 102 in Figure 9 are connected to each other via USB cable 115 to enable communication. USB-equipped device 102 has error detecting means 104 and voltage changing means 105. When error detecting means 104

Fig.1



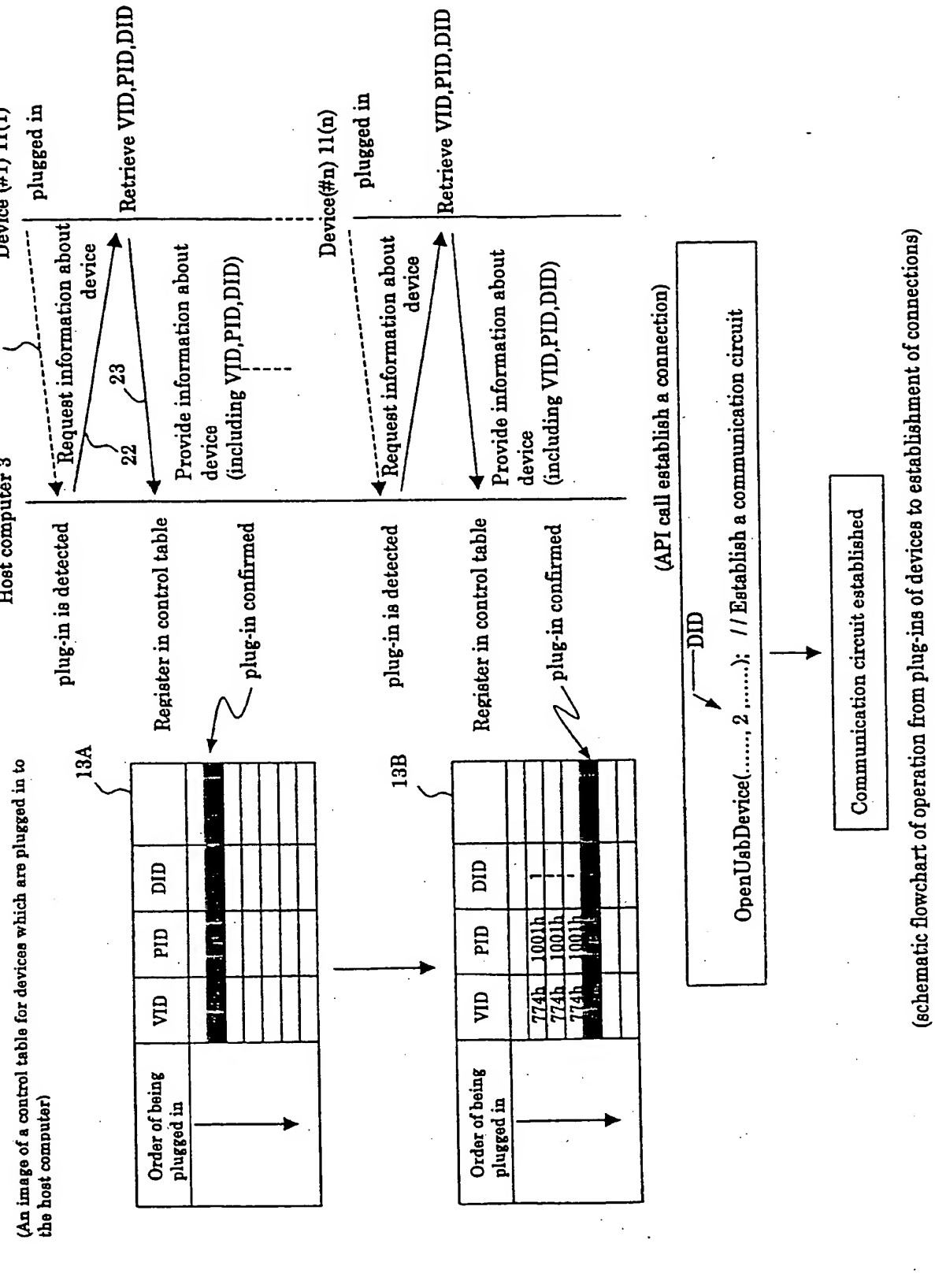
(components of a device required to realize the present invention)

Fig.2

```
int   OpenUsb Device(
      ________
      ________
      int   DeviceID,    //   arguments to assign a DID
      ________
      ________
      );
```

(Schematic configuration of an API to establish a connection)

Fig.4



(schematic flowchart of operation from plug-ins of devices to establishment of connections)